3-7-2006

Optimization of online data integration

J. R. Getta
University of Wollongong, jrg@uow.edu.au

Optimization of online data integration

Abstract

Online data integration is a process of continuous consolidation of data transmitted over the wide area networks with data already stored at a central site of a multidatabase system. The continuity of the process requires activation of data integration procedure each time a new portion of data is received at a central site. Efficient implementation of online data integration needs a new system of elementary operations on the increments and/or decrements of data and the intermediate results of integration. This work shows how to derive a new system of elementary operations for online data integration from a system of base operations on the data containers. In particular, we define a new system of online operations based on the system of binary operations of relational algebra. The paper analyses the properties of the new system and describes the transformations of global data integration expressions into the collections of online data integration plans. It is presented how the system can be used for the comprehensive analysis and optimization of online data integration plans. The optimization techniques described in the paper include reduction of input data increments, identification and elimination of intermediate materializations, and reduction of fixed size arguments in online data integration plans.

Disciplines

Physical Sciences and Mathematics

Publication Details

This article was originally published as: Getta, JR, Optimization of online data integration, 7th International Baltic Conference on Databases and Information Systems 2006, 3-6 July 2006, 91-97. Copyright 2006 IEEE.

Optimization of Online Data Integration

Janusz R. Getta
School of Information Technology and Computer Science
University of Wollongong
Wollongong, NSW 2522, Australia

Abstract— **Online data integration is a process of continuous consolidation of data transmitted over the wide area networks with data already stored at a central site of a multidatabase system. The continuity of the process requires activation of data integration procedure each time a new portion of data is received at a central site. Efficient implementation of online data integration needs a new system of elementary operations on the increments and/or decrements of data and the intermediate results of integration.**

This work shows how to derive a new system of elementary operations for online data integration from a system of base operations on the data containers. In particular, we define a new system of online operations based on the system of binary operations of relational algebra. The paper analyses the properties of the new system and describes the transformations of global data integration expressions into the collections of online data integration plans. It is presented how the system can be used for the comprehensive analysis and optimization of online data integration plans. The optimization techniques described in the paper include reduction of input data increments, identification and elimination of intermediate materializations, and reduction of fixed size arguments in online data integration plans.

I. Introduction

One of the central problems in the development of global information systems is an efficient *integration of data* transmitted over the wide area networks. A *data integration subsystem* is one of the main components of a multidatabase system that provides the users with a transparent and centralized view of several distributed and heterogeneous databases. *Online data integration* is a process of continuous consolidation of data transmitted over a network with data already available at a central site of a multidatabase system. The intermediate results of online data integration provide a user with the most up-to-date results of a query being processed by the system. Online integration of data does not delay the processing of incoming data until all transmissions from the remote sites are completed. Instead, the transmitted packets of data are integrated with the partial results as soon as they arrive at a central site. Such approach reduces time spent by a user waiting for the first results from a running application and it allows for an early termination of an application when the initial results are inconsistent with the expectations. An important advantage of online data integration is its ability to utilise the unused computational resources at a central site while data are transmitted over a network. The recent forecasts [1], [2] anticipate that freely available distributed data sets and fast wide-area networks will promote online data integration as an important research area of the distributed computing and financial data processing.

The performance of online data integration depends on the advanced *online algorithms* used for consolidation of data and on the efficient optimization of online data integration plans. *Online algorithms* [3] process the increments of input data sets and continuously improve the results when the new increments are available for processing. A virtual memory manager or a scheduler of database transactions are the typical implementations of online algorithms.

The research works on online data integration can be traced back to the research on query processing in multidatabase or federated database systems [4], [5]. Unpredictable behaviour of the data transmissions in the wide area networks and strong autonomy of the remote database systems makes the estimation of query processing time hard and imprecise. It makes a query processing plan optimal on one occasion, ineffective on the other. The external factors affecting the performance of query processing in multidatabase systems promote the *reactive query processing techniques*. Reactive processing of a query starts from a hypothetically optimal plan, and whenever the further processing is impossible due to the network problems, or unavailability of data from an external database site, the processing is continued accordingly to a modified plan.

The early works on the reactive query processing techniques are based on *partitioning* [6], [7] and *dynamic modification of query processing plans* [8], [9], [10]. Partitioning decomposes a query execution plan into sub-plans at a point when the further computations are no longer possible due to a lack of data. A dynamic modification technique finds a plan equivalent to the original one and such that it is possible to continue integration of the available data sets. The similar approaches dynamically change an order in which the join operations are executed depending on the arguments available at a central site. These techniques include *query scrambling* [11], [12] and *dynamic scheduling of operators* [13].

Another important research direction in reactive query processing aims at the optimization of relational algebra operations used for the data integration. These works include the new versions of join operation customised to online query processing, e.g. *pipelined join operator XJoin* [14], *ripple join* [15], *double pipelined join* [16], and *hash-merge join* [17].

A technique of *redundant computations* simultaneously pro-

1-4244-0345-6/06/$20.00

cesses a number of data integration plans leaving a plan that that provides the most advanced results [18].

The solutions based on data partitioning integrate the different components of integrated arguments accordingly to the different plans. The *Eddies* are able to process each tuple accordingly to a different plan [19]. A concept of *state modules* described in [20] allows for concurrent processing of the tuples and dynamically divides data integration task among different plans and executes the plans sequentially or in parallel. *Adaptive data partitioning* [21] technique processes different partitions of the same argument using different data integration plans. The works on adaptive data partitioning [21] and optimizations of data stream processing [22] were the first attempts to use the associativity of join operation to integrate the separate partitions of the same arguments with the different integration plans. An adaptive and online processing of data integration plans proposed in [23] also concentrates on the sets of elementary operations for data integration and proposes an algorithm that finds the best integration plan for the outcomes of the most recent transmissions.

It is interesting that many of the recently developed techniques for *data stream processing* [24], [22] can be reused for online data integration. It is so because the implementations of both groups of systems require online algorithms that efficiently recompute a query each time one of the data containers processed by the query is changed.

The works [25], [26], [27], [28] review the most important data integration techniques proposed so far. A more up-to-date and more detailed overview of the past works on adaptive data integration can be found in [29].

The data integration techniques reviewed above use the relational model of data as a global data model of a multidatabase system and the relational algebra as a language for the formal specification of data integration plans. Unfortunately, any system of operations on the fixed size data containers, like for example a system that includes join, antijoin and set union operations of the relational algebra, is not suitable for the representation of online data integration procedures. Online data integration needs the operations whose at least one argument is an increment or decrement of data container, the other arguments is are data containers, and whose result is an increment or decrement of data container as well. In such a system, the computations start from an operation on an increment of data container transmitted through a network and the data containers located at a central site. Then, an increment or decrement of data container returned by the first operation becomes an argument of the second operation. Each next operation uses an increment or decrement of data container produced by the previous operation and also returns an increment or decrement of data container for the successive operation. Finally, an increment or decrement of data obtained at the end of this process is integrated with the results of the previous integration cycle providing the most up-to-date evaluation of a query.

As the relational model is typically used as a global data model of multidatabase system, the systems of elementary operations for online data integration must also act on the increments and decrements of relational tables and already integrated contents of the relational tables. In the consequence, the elementary operations of online data integrator should process the increments or decrements of relational tables against the fixed size relational tables. This work shows how to derive a system of elementary operations on the increments or decrements of data containers from the system of base operations on data containers and how create and how to optimize the online integration plans formed from the sequences of the new operations.

Typically, a query submitted by a user of a multidatabase system is translated into a relational algebra expression whose arguments are the results of sub-queries processed at the local database sites. The expression, called as a *data integration expression*, determines a way how the results of sub-queries are combined into the final answer. An online evaluation of a data integration expression requires its transformation into a sequence of elementary operations on the increments or decrements of data containers. Such a sequence of elementary operations is called as an *online integration plan*. An online integration plan integrates an increment of the results delivered by a sub-query with the results already available at a central site. The total number of online integration plans required for a given data integration expressions is determined by the total number of its arguments where at least one online plan is needed for each one of the arguments. The increments of an argument of a data integration expression may be processed in many different ways because usually there exist many online integration plans for the argument. Selection of the the best online integration plan from a set of equivalent plans is one of the optimization strategies considered in this work.

The paper is organized in the following way. We start from the presentation of a sample data integration model in Section II. Section III shows how to derive a system of elementary operations on the increments or decrements of data containers. The sample derivations of elementary operations from the base operations of relational algebra are presented in Section IV. The next section (V) defines an online integration plan and shows how to transform a data integration expression into a set of online plans. Three optimizations techniques for online integration plans are presented Section VI. Finally, Section VII summarises and concludes the paper.

II. Data Integration Model

In this work we consider a multidatabase system that integrates a number of distributed and heterogeneous database systems such that the remote database sites are entirely transparent at a central site. A middleware implementing the system provides a user with a view of a homogeneous database that consists of the data containers $r_1, \ldots, r_m$. If the relational database model is used as a global data model of the system then the containers are the relational tables.

A query $q(r_1, \ldots, r_k)$ submitted by a user is decomposed into k sub-queries $q_{r_1}, \ldots, q_{r_k}$ that encapsulate the computations performed at the remote database systems in order to

materialize the data containers $r_1, \ldots, r_m$ at a central site. Two generic strategies of distributed query processing either optimize an overall amount of time spend on the computations or optimize the total amount of data transmitted over a network. Query processing time is minimized when the sub-queries $q_{r_1}, \ldots, q_{r_k}$ are submitted and processed simultaneously at the remote sites. Processing of sub-queries one at a time and applying the results of one sub-query to modify the remaining sub-queries minimizes the amounts of transmitted data. A continuum of hybrid distributed query processing strategies is included between these two generic strategies. Selection of the best strategy is a hard problem and it is beyond a scope of this paper.

This work applies a strategy that minimizes query processing time through the simultaneous computations of sub-queries at the remote database sites and simultaneous transmissions of the partial results to a central site. The partial results are transformed into the containers $r_1, \ldots, r_k$, which are structurally consistent with a global data model of a multidatabase system, e.g. into the relational tables. The partial results are integrated into the final answer accordingly to an *integration plan* $\mathcal{P}(r_1, \ldots, r_k)$ derived from the original query q and built from the base operations on the data containers, e.g. the relational algebra operations on the relational tables.

A quite ineffective and naive approach would be to delay the integration until all partial results are entirely transmitted to a central site. Instead, whenever the computational resources are available, it is possible to start and to continue the integration each time a new packet of data is received at a central site. Such technique, known as *online data integration*, makes the process of integration more flexible because there is no need to wait for the entire arguments when a data integration plan is evaluated accordingly to a given order of operations. Online data integration finds the intermediate results of query processing each time a new portion of data is received at a central site and no matter which argument it belongs to. This idea invalidates a concept of a single data integration plan because the continuous re-evaluation of entire plan each time a new packet of data is received takes too much time. Instead, a data integration plan is transformed into a set of *online data integration plans* where each plan represents an integration procedure for the increments of one argument. An online plan is a sequence of so called *id-operations* on the increments or decrements of data containers and other fixed size containers.

III. ID-Operations

Let $r_1, \ldots, r_k$ be the data containers, e.g relational tables. A *base operation* A is an operation whose arguments are the containers r_i, r_j and whose result is a data container, e.g. a join of relational tables is a base operation of the relational algebra.

A *modification* δ_i of a data container r_i is a pair of data containers $<\delta_i^-, \delta_i^+>$ such that $\delta_i^- \cap r_i = \delta_i^-$ and $\delta_i^+ \cap r_i = \emptyset$. An operation that applies a modification δ_i to a data container r_i is denoted by $r_i \oplus \delta_i$ and it is called as an *integration operation*. In the relational database model the integration of a modification δ_i with a relational table r is defined as $(r - \delta_i^-) \cup \delta_i^+$ where r and both components of δ_i have the same schemas. Then, δ_i^- is a set of rows that have to be removed from r and δ_i^+ is a set of rows that have to be added to r in order to implement a modification δ_i.

An *incremental/decremental operation* later on called as an *id-operation* of the first argument r_i of a base operation $A(r_i, r_j)$ is denoted by $\alpha_A(\delta_i, r_j)$ and it is defined as the smallest modification δ_A that should be integrated with the result of $A(r_i, r_j)$ to obtain the result of $A(r_i \oplus \delta_i, r_j)$. In the other words, the result of id-operation $\alpha_A(\delta_i, r_j)$ is the smallest solution of an equation (1) below.

$$A(r_i, r_j) \oplus \alpha_A(\delta_i, r_j) = A(r_i \oplus \delta_i, r_j) \tag{1}$$

An id-operation of the second argument of a base operation $A(r_i, r_j)$ is denoted by $\beta_A(r_i, \delta_j)$ and it is defined in a similar way. A result of id-operation $\beta_A(r_i, \delta_j)$ is the smallest solution of an equation (2) below.

$$A(r_i, r_j) \oplus \beta_A(r_i, \delta_j) = A(r_i, r_j \oplus \delta_j) \tag{2}$$

The id-operations allow for the fast re-computation of a base operation after one of its arguments has changed. An idea is to apply an id-operation to the modification and the other argument of the base operation to get a modification that can be integrated with the old result of base operation to obtain the new result. Such procedure is faster because one of the arguments of id-operation and data integration operation on the left hand sides of the equations (1) and (2) is small enough to be always kept in a transient memory. The sequences of id-operations allow for the implementation of online data integration plans were a modification processed by an id-operation returns a modification processed by the next id-operation and so on until the last id-operation in the plan returns a modification that updates the previous result of integration.

The analytical solutions of the equations (1) and (2) provide the explicit definitions of *id-operations*. Both equations are the set algebra equations of type $P \oplus x = Q$. To find the smallest solution of an equation of this type we transform it into a fixpoint equation (3).

$$x = x \cup (((P \oplus x) - Q) \cup (Q - (P \oplus x))) \tag{3}$$

A fixpoint equation above can be solved with the Kleene fixpoint theorem, which states that for any complete lattice L and a monotone function $F : L \to L$ the least fixed point of F is equal to $\cup_{n \in N} F^n(\perp)$ where $\perp$ is the smallest element in a lattice L. Assuming that $x \cup (((P \oplus x) - Q) \cup (Q - (P \oplus x)))$ if monotone the first iteration for $\perp = \emptyset$ provides $x^{(0)} = (P - Q) \cup (Q - P) = P \div Q$. We reach a fixed point in the second iteration where for $x^{(1)} = (P - Q) \cup (Q - P) = P \div Q$ we get the same result. Therefore, the solutions of the equations (1) and (2) are as follows.

$$\alpha_A(\delta_i, r_j) = A(r_i, r_j) \div A(r_i \oplus \delta_i, r_j) \tag{4}$$

$$\beta_A(r_i, \delta_j) = A(r_i, r_j) \div A(r_i, r_j \oplus \delta_j) \tag{5}$$

IV. Relational Algebra Based id-Operations

This section shows how to derive the id-operations from the base operations of the relational algebra. We consider the binary operations of union ($\cup$), join ($\bowtie$), and antijon ($\sim$) and we assume that selection operation is always directly applied to the arguments of binary operations and projection is applied only one time to the final result of query processing. After the replacement of $A(r,s)$ with $r \cup s$ and $r \oplus \delta_i$ with $(r - \delta_i^-) \cup \delta_i^+$ in the equations (4) and (5) we obtain the equations below.

$$\alpha_\cup(\delta_i, r_j) = (r_i \cup r_j) \div (((r_i - \delta_i^-) \cup \delta_i^+) \cup r_j) \quad (6)$$

$$\beta_\cup(r_i, \delta_j) = (r_i \cup r_j) \div (r_i \cup ((r_j - \delta_j^-) \cup \delta_j^+)) \quad (7)$$

Next, we separately solve the equations (6) and (7) for the negative (δ_i^-) and the positive (δ_i^+) components of a modification δ_i.

$$\alpha_\cup(\delta_i, r_j) =< (\delta_i^- \cap r_i) - r_j, (\delta_i^+ - r_i) - r_j > \quad (8)$$

It is possible to simplify the result above using the properties of a modification δ_i, i.e. δ_i^+ disjoint with r_i and δ_i^- included in r_i.

$$\alpha_\cup(\delta_i, r_j) =< \delta_i^- - r_j, \delta_i^+ - r_j > \quad (9)$$

An operation $\beta_\cup(r_i, \delta_j)$ can be derived in the same way. If a base operation is commutative then the respective id-operations for both arguments are always the same.

$$\beta_\cup(r_i, \delta_j) =< \delta_j^- - r_i, \delta_j^+ - r_i > \quad (10)$$

The implementations of id-operations (9) and (10) mean that whenever one of the arguments of set union operation is modified then it is possible to evaluate $\delta_i^+ - r_j$ and $\delta_i^- - r_j$ and to integrate the results with the previous result of set union to get a new result.

The id-operations of a base operation of relational join ($\bowtie$) can be found in the same way from the equations (4) and (5).

$$\alpha_\bowtie(\delta_i, r_j) =< \delta_i^- \bowtie r_j, \delta_i^+ \bowtie r_j > \quad (11)$$

$$\beta_\bowtie(r_i, \delta_j) =< r_i \bowtie \delta_j^-, r_i \bowtie \delta_j^+ > \quad (12)$$

The id-operations $\alpha_\sim(\delta_i, r_j)$ can be found in the same way from the equations (4) and (5).

$$\alpha_\sim(\delta_i, r_j) =< \delta_i^- \sim r_j, \delta_i^+ \sim r_j > \quad (13)$$

The id-operations $\beta_\sim(r_i, \delta_j)$ must be derived in a different way because the replacement of a base operation $A(r_i, r_j)$ with $r_i \sim r_j$ makes the right hand side of a fixpoint equation non-monotone. Here, a base operation $A(r_i, r_j)$ is replaced with $r_i \sim r_j$ in the original equation (1).

$$(r_i \sim r_j) \oplus \beta_\sim(r_i, \delta_j) = r_i \sim (r_j \oplus \delta_j) \quad (14)$$

Then, like in the previous cases, we separately consider the incremental and decremental components of a modification δ_j.

$$(r_i \sim r_j) \cup \beta_\sim(r_i, \delta_j) = r_i \sim (r_j \cup \delta_j^+) \quad (15)$$

It is equivalent to

$$(r_i \sim r_j) \cup \beta_\sim(r_i, \delta_j) = (r_i \sim r_j) - (r_i \ltimes \delta_j^+) \quad (16)$$

Therefore, an incremental component of δ_j contributes to a decremental component of $\beta_\sim(r_i, \delta_j)$. In the same way, we find a decremental component of δ_j that contributes to an incremental component of $\beta_\sim(r_i, \delta_j)$.

$$\beta_\sim(r_i, \delta_j) =< r_i \ltimes \delta_j^+, r_i \ltimes \delta_j^- > \quad (17)$$

As a simple application of the id-operations derived above, consider the processing of a global data integration plan $q(r,s,t) = t \bowtie (r - s)$ after a modification $\delta_s =< \emptyset, \delta_s^+ >$ is applied to an argument s. Then, the equations (12) and (17) contribute to the following expressions: $\delta_{rs} = \beta_\sim(r, \delta_s) =< r \ltimes \delta_s^+, \emptyset >$ and $\delta_{rst} = \beta_\bowtie(t, \delta_{rs}) =< t \bowtie (r \ltimes \delta_s^+), \emptyset >$. Therefore, a modification $\delta_s =< \emptyset, \delta_s^+ >$ of an argument s needs the following modification of the current result of integration $q'(r,s,t) = q(r,s,t) - t \bowtie (r \ltimes \delta_s^+)$.

Note, that processing of a modification δ_t of an argument t needs either the materializations of intermediate results of a subexpression $(r - s)$ or the transformation of a global integration plan $q(r,s,t) = t \bowtie (r - s)$ into an equivalent form where its syntax tree is either left (right)-deep and an argument t is in the leftmost (rightmost) position of the tree.

V. Online Integration Plans

An *online integration plan* is a sequence $r_0{:}\gamma_1(r_1)\ldots\gamma_n(r_n)$ where r_0 is a data container whose modification is processed accordingly to the plan and each $\gamma_i(r_i)$, is either an abbreviation of id-operation $\alpha(\delta_{r_j}, r_i)$ or $\beta(r_j, \delta_{r_i})$ or an abbreviation of id-operation $\delta_{r_j} \oplus r_i$. The evaluation of an online plan starts from the evaluation of the first id-operation $\alpha_1(\delta_{r_0}, r_1)$ or $\beta_1(r_1, \delta_{r_0})$. Then, a modification δ_{α_1} produced by the first id-operation becomes an argument of the second id-operation $\alpha_2(\delta_{\alpha_1}, r_2)$ or $\beta_2(r_2, \delta_{\alpha_1})$ and so on. The adjacent id-operations in an online plan always pass the results produced by id-operation γ_i as the arguments of the next id-operation γ_{i+1}. For example, an online plan $r{:}\alpha_\bowtie(s)\alpha_-(t) \oplus (w)$ represents a sequence of computations that starts from $\delta_{rs}{:=}\alpha_\bowtie(\delta_r, s)$, where δ_r is a modification of an argument r, through $\delta_{rst}{:=}\alpha_-(\delta_{rs}, t)$ and finally $w{:=}w \oplus \delta_{rst}$.

The online plans are derived from a data integration plan $\mathcal{P}(r_1, \ldots, r_k)$ constructed at the early stages of query processing. An online plan for an argument r_i of $\mathcal{P}(r_1, \ldots, r_k)$ is constructed by the traversal of a syntax tree of $\mathcal{P}$ from a leaf node labelled by r_i to the root node. Initially, at a leaf node r_i, we form an empty expression $r_i{:}$. A procedure `Traverse(i,j)` traverses a syntax tree of from node `i` to node `j`. and it is as repeated as many times as it is needed to reach the root node. The procedure performs the following actions.

If a node `i` is a leaf node r_i then
  if a node `k` (see Fig. 1) is a leaf node r_k then
    append $\gamma_j(r_k)$ to an online integration plan
    where γ_j is an id-operation of an operation at
    a node `j`
  else if a node `k` is a root of a nonempty subtree then
    append $\gamma_j(m_{jk})$ to an online integration plan
    where m_{jk} is a materialization of

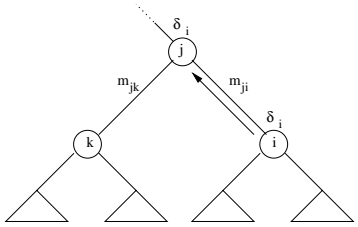

Fig. 1. Syntax tree traversed by `Traverse(i,j)`.

the computations represented by a subtree with a root node `k`

else if a node `i` is a root node of a nonempty subtree then
  if a node `k` (see Fig. 1) is a leaf node r_k then
    append $\gamma_j(r_k)$ to an online data integration plan
  else if a node `k` is a root node of a nonempty subtree then
    append $\gamma_j(m_{jk})$ to an online integration plan
  append $\oplus m_{ji}$ to an online integration plan
end.

When a root node is reached by `Traverse(i,j)` then a data integration operation $\oplus w$ is appended to an online data integration plan where w is the final result of data integration.

For example, the procedure described above generates the following online integration plans when applied to a data integration expression $r - (s \bowtie t)$:
r: $\alpha_-(w_{st})\ \oplus(w)$
s: $\alpha_{\bowtie}(t)\ \oplus(w_{st})\ \beta_-(r)\ \oplus(w)$
t: $\beta_{\bowtie}(s)\ \oplus(w_{st})\ \beta_-(r)\ \oplus(w)$
The online plans listed above are the sequences of id-operations and data integration operations that process the increments of the arguments r, s, and t in order to obtain the most up-to-date evaluation of data integration expression.

VI. Optimization of Online Integration Plans

Like in the traditional query processing, optimization of online data integration transforms the online plans into the semantically equivalent plans that can be implemented more efficiently. The first group of transformations reduces the partial results transmitted over a network by moving the most restrictive id-operations towards the beginning of online plans. The other group of transformations removes the intermediate materializations in order to concatenate the adjacent id-operations and to eliminate the updates of intermediate materializations. The last group of transformations reduces the size of selected arguments when the transmissions of the other arguments are completed. All optimization techniques presented in this work are the cost based optimizations. A transformed online plan is considered to be more optimal when its implementation is less expensive than the implementation the same plan before the transformation. A near optimal plan is obtained through the systematic transformations of online plans towards the plans with the lower implementation costs.

A. Reduction of Modifications

An id-operation is restrictive when the modifications returned by the operation are smaller than its input modifications. For example, an id-operation is restrictive when a unary selection is applied to the input modifications processed by the operation. Moving the most restrictive id-operations towards the beginning of online data integration plan reduces the amount of data processed by the remaining operations. An order of two adjacent id-operations can be changed when the operations are commutative. For instance, a sequence of operations $\alpha_{\bowtie}(s)$ $\alpha_{\bowtie}(t)$ is semantically equivalent to a sequence $\alpha_{\bowtie}(t)$ $\alpha_{\bowtie}(s)$ because $\alpha_{\bowtie}$ operation is commutative. Commutativity of id-operations is determined by the associativity of the relational algebra operations that implement the respective id-operations. In the example above, an expression $(\delta \bowtie s) \bowtie t$ implements a sequence of id-operations $\alpha_{\bowtie}(s)$ $\alpha_{\bowtie}(t)$. The expression is equivalent to $(\delta \bowtie t) \bowtie s$, which is an implementation of online plan $\alpha_{\bowtie}(t)$ $\alpha_{\bowtie}(s)$. It is possible to show that the following pairs of id-operations are commutative: $\alpha_{\cup}(s)$ $\alpha_{\cup}(t)$, $\alpha_{\sim}(s)$ $\alpha_{\sim}(t)$, $\beta_{\sim}(s)$ $\beta_{\sim}(t)$, $\alpha_{\sim}(s)$ $\alpha_{\bowtie}(t)$, $\alpha_{\cup}(s)$ $\alpha_{\sim}(t)$, $\alpha_{\sim}(s)$ $\alpha_{\cup}(t)$, and $\beta_{\sim}(s)$ $\beta_{\cup}(t)$.

In some cases, a modification can be reduced by an additional operation executed before a given sequence of id-operations. For example, a sequence of operations $\alpha_{\cup}(t)$ $\beta_{\sim}(s)$ implemented as a relational algebra expression $s \ltimes (\delta - t)$ can be transformed into $s \ltimes ((\delta \ltimes s) - t)$, which implements an operation $\delta \ltimes s$ followed by the original sequence of id-operations $\alpha_{\cup}(t)$ $\beta_{\sim}(s)$. The other cases of this kind include the pairs of id-operations $\alpha_{\bowtie}(t)\beta_{\sim}(s)$ and $\beta_{\sim}(t)\alpha_{\bowtie}(s)$, which are equivalent to $\delta \ltimes s$ followed by the original pairs of id-operations.

B. Elimination of Materializations

The procedure generating the online integration plans creates the intermediate materializations whenever an operation in a data integration plans acts on the results of two subexpressions. The materializations have a negative impact on performance because of the additional data integration operation required in the online plans. For example, the translation of a data integration plan $(r \bowtie s) \bowtie t$ provides the online plans where the additional data integration operation $\oplus(w_{rs})$ must be inserted into the plans for processing the increments of r and s.
r: $\alpha_{\bowtie}(s)\ \oplus(w_{rs})\ \alpha_{\bowtie}(t)\ \oplus(w)$,
s: $\beta_{\bowtie}(r)\ \oplus(w_{rs})\ \alpha_{\bowtie}(t)\ \oplus(w)$,
t: $\beta_{\bowtie}(w_{rs})\ \oplus(w)$.

The materializations can be completely eliminated if it is possible to transform a data integration expressions to left (right)-deep syntax tree for each one of the arguments of the expression being in the left (right)-deep corner of the tree. For example, to eliminate a materialization w_{rs} from the online plans given above it is enough to transform the original data integration plan into $(r \bowtie t) \bowtie s$ and to find a new online integration plan for processing the increments of an argument t. Then $\oplus(w_{rs})$ can be removed from the remaining online

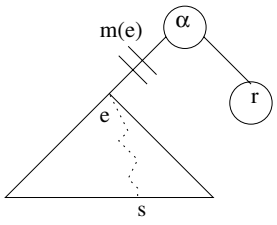

Fig. 2. A reduction of s by a complete argument r.

plans because a materialization w_{rs} is no longer used in an online plan for an argument t, see below.
r: $\alpha_{\bowtie}(s)\alpha_{\bowtie}(t)\ \oplus(w)$,
s: $\beta_{\bowtie}(r)\alpha_{\bowtie}(t)\ \oplus(w)$,
t: $\beta_{\bowtie}(r)\alpha_{\bowtie}(s)\ \oplus(w)$.

A materialization can be removed when an id-operation is distributive over the operation of relational algebra expression that implements the materializations. In the example above it is true that after the replacement of materialization w_{rs} with its implementation $r \bowtie s$ an implementation of operation $\beta_{\bowtie}(r \bowtie s) = (r \bowtie s) \bowtie \delta_t$. Then due to the associativity of join operation it is equivalent to $(r \bowtie \delta_t) \bowtie s$, which is an online plan $\beta_{\bowtie}(r)\alpha_{\bowtie}(s)$ that does not use a materialization w_{rs}. Distributivity of id-operations over the operations of relational algebra holds in the following cases $\alpha_{\bowtie}(s \bowtie t)$, $\alpha_{\bowtie}(s \cup t)$, $\alpha_{\sim}(s \sim t)$, $\alpha_{\sim}(s \cup t)$, $\beta_{\sim}(s \sim t)$.

C. Reduction of Arguments

It is possible to reduce the size of the arguments of data integration plan when the transmissions of the other arguments are completed. Consider a skeleton of data integration plan given in Fig. 2 and assume that the schemas of r and s have at least one common attribute whose value is used by an operation α. Next, assume that the transmission of an argument r is completed. Then it may happen that the result of $\alpha(m(e), r)$ where $m(e)$ is a materialization of subtree e with an argument s is the same as the result of $\alpha(m(e'), r$ where $m(e')$ is a materialization of subtree e with an argument $s' \subseteq s$. It means that certain rows included in s do not contribute to the result of operation α due to the contents of an argument r. Then, it is possible to remove such rows from s. As an example consider a data integration expression $r \sim (s \bowtie_b t)$ and assume that an attribute b is common to the schemas of r, s, and t. If the transmission of an argument r is completed then an intermediate result of $s \bowtie t$ may contain the rows that have no impact on the result antijoin operation. Then, it is possible to remove from r and s all rows whose values of an attribute b are not included in a set of values of attribute b in an argument r. To eliminate such rows the arguments s and t should be reduced to $s \ltimes_b r$ and $t \ltimes_b r$. and online integration plans for r and s should be transformed into:
s: $\ltimes_b(r)\ \alpha_{\bowtie}(t)\ \beta_{-}(r)\ \oplus(w)$
t: $\ltimes_b(r)\ \beta_{\bowtie}(s)\ \beta_{-}(r)\ \oplus(w)$.

In a general case, the reductions are possible when the following conditions are satisfied.

1) The schemas of arguments s and r have a set of common attributes x and $x \subseteq y$ where y is a set attributes used by operation α, e.g. join attributes of join operation.
2) A path from argument s to $m(e)$ never passes through the right argument of antijoin $(\sim)$, e.g. an operation $t \sim s$ invalides the condition
3) Operation α is join operation $(\bowtie)$ or operation α is antijoin operation $(\sim)$ where an argument r must be the left argument of antijoin, i.e. $r \sim m(e)$.

If the conditions above are satisfied then it is possible to reduce an argument s to $s \ltimes_x r$ and to transform the respective online plan s:$\gamma_1(r_1)\ldots\gamma_n(r_n)$ into s: $\ltimes_x(r)\ \gamma_1(r_1)\ldots\gamma_n(r_n)$.

VII. Summary, Conclusions, and Future Work

This work considers the optimization of online data integration in a multidatabase database system. A process of online data integration continuously appends the increments of data sets transmitted over a network to the data sets already available at a central site and recomputes a data integration expression after each append. To make this process efficient we propose a new system of operations for processing the increments of data sets against already transmitted data. The paper introduces a concept of id-operation that encapsulates the elementary computations on the increments and/or decrements of data. Next, we show how to derive a system of id-operations from a system of relational algebra operations and how to transform the data integration expressions into the sets of online integration plans composed of the sequences of id-operations and data integration operations. Finally, the paper considers three types of optimization techniques for online integration plans. These techniques include reduction of input data, elimination of materializations and reduction of arguments.

Several interesting problems remain open. These include the questions whether the materializations can always be eliminated from the online integration plans and when it is beneficial to do so. Another interesting problem is identification of all reductions that are possible in a given moment of time and scheduling of the reductions in a process of online data integration. The other problems include the derivations of more sophisticated systems of id-operations from the systems of binary operations different from the relational algebra e.g. a system including aggregation operations, further investigations of the properties of online integration plans and more advanced data integration algorithms where the application of a particular online plan depends on what increments of data are available at the moment.

References

[1] I. Foster and R. L. Grossman, "Data integration in a bandwidth-rich world," *Communications of the ACM*, vol. 46, pp. 51–57, Nov. 1996.
[2] A. Pan and Á. Vina, "An Alternative architecture for financial data integration," *Communications of the ACM*, vol. 47, pp. 37–40, May 2004.
[3] A. Fiat and G. J. Woeginger, *On Line Algorithms, The State of the Art.* Springer Verlag, 1998.

[4] V. Srinivasan and M. J. Carey, "Compensation-based on-line query processing," in *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data,* 1992, pp. 331-340.

[5] F. Ozcan, S. Nural, P. Koksal, C. Evrendilek, and A. Dogac, Dynamic query optimization in multidatabases, *Bulletin of the Technical Committee on Data Engineering,* vol. 20, pp. 38-45, March 1997.

[6] R. L. Cole and G. Graefe, Optimization of dynamic query evaluation plans, in *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, 1994, pp. 150–160.

[7] N. Kabra and D. J. DeWitt, Efficient mid-query re-optimization of sub-optimal query execution plans, in *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, 1998, pp. 106–117.

[8] J. Chudziak and J. R. Getta, On efficient query evaluation in multi-database systems, in *Proceedings of the Second International Workshop on Advances in Database and Information Systems, ADBIS95*, 1995, pp. 46-54.

[9] J. R. Getta and S. Sedighi, Optimizing global query processing plans in heterogeneous and distributed multi database systems, in *Proceedings of the 10th Intl. Workshop on Database and Expert Systems Applications, DEXA'99,* 1999, pp. 12-16.

[10] J. R. Getta, Query scrambling in distributed multidatabase systems, in *Proceedings of the 11th Intl. Workshop on Database and Expert Systems Applications, DEXA' 2000,* 2000, pp. 647–652.

[11] T. Urhan, M. J. Franklin, and L. Amsaleg, "Cost based query scrambling for initial delays," in *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data,* 1998, pp. 130–141.

[12] L. Amsaleg, J. Franklin, and A. Tomasic, "Dynamic query operator scheduling for wide-area remote access," *Journal of Distributed and Parallel Databases,* vol. 6, pp. 217–246, 1998.

[13] T. Urhan and M. J. Franklin, "Dynamic pipeline scheduling for improving interactive performance of online queries," in *Proceedings of International Conference on Very Large Databases VLDB'01,* 2001.

[14] T. Urhan and M. J. Franklin, "Xjoin: a reactively-scheduled pipelined join operator," *IEEE Data Engineering Bulletin,* vol. 23, pp. 27–33, June 2000.

[15] P. J. Haas and J. M. Hellerstein, "Ripple joins for online aggregation," in *Proceedings of the 1999 ACM SIGMOD Intl. Conf. on Management of Data,* 1999, pp. 287–298.

[16] Z. G. Ives, D. Florescu, M. Friedman, A. Y. Levy, and D. S. Weld, "An adaptive query execution system for data integration," in *Proceedings of the 1999 ACM SIGMOD Intl. Conf. on Management of Data,* 1999, pp. 299–310.

[17] M. F. Mokbel, M. Lu, and W. G. Aref, "Hash-merge join: a nonblocking join algorithm for producing fast and early join results," in *Proceedings of the 20th International Conference on Data Engineering ICDE2004,* 2004, pp. 251-263.

[18] G. Antoshenkov and M. Ziauddin, "Query processing and optimization in Oracle Rdb," *VLDB Journal,* vol. 5, pp. 229–237, Dec. 2000.

[19] R. Avnur and J. M. Hellerstein, "Eddies: continuously adaptive query processing," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data,* 2000, pp. 261–272.

[20] V. Raman, A. Deshpande, and J. M. Hellerstein, "Using state modules for adaptive query processing," in *Proceeding of International Conference on Management of Data,* 2003, pp. 353–366.

[21] Z. G. Ives, A. Y. Halevy, and D. S. Weld, "Adapting to source properties in processing data integration queries," in *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data,* 2004, pp. 395–406.

[22] J. R. Getta and E. Vossough, "Optimization of data stream processing," *SIGMOD Record,* vol. 33, pp. 34–39, Sept. 2004.

[23] J. R. Getta, "On adaptive and online data integration," in *Proceedings of the Intl. Workshop on Self-Managing Database Systems, 21st Intl. Conf. on Data Engineering, ICDE'05,* 2005, pp. 1212–1220.

[24] S. Madden, M. A. Shah, J. M. Hellerstein, and V. Raman, "Continuously adaptive continuous queries over streams," in *Proceedings of the 2002 ACM SIGMOD Intl. Conf. on Management of Data,* 2002, pp. 49–60.

[25] L. Bouganim, F. Fabret, and C. Mohan, "A dynamic query processing architecture for data integration systems," *Bulletin of the Technical Committee on Data Engineering,* vol. 23, pp. 42–48, June 2000.

[26] G. Graefe, "Dynamic query evaluation plans: some course corrections?" *Bulletin of the Technical Committee on Data Engineering,* vol. 23, pp. 3–6, June 2000.

[27] J. M. Hellerstein, M. J. Franklin, S. Chandrasekaran, A. Deshpande, K. Hildrum, S. Madden, V. Raman, and M. A. Shah, "Adaptive query processing: Technology in evolution," *Bulletin of the Technical Committee on Data Engineering,* vol. 23, pp. 7–18, June 2000.

[28] Z. G. Ives, A. Y. Levy, D. S. Weld, D. Florescu, and M. Friedman, "Adaptive query processing for internet applications," *Bulletin of the Technical Committee on Data Engineering,* vol. 23, pp. 19–26, June 2000.

[29] A. Gounaris, N. W. Paton, A. A. Fernandes, and R. Sakellariou, "Adaptive query processing: a survey," in *Proceedings of 19th British National Conference on Databases,* 2002, pp. 11–25.